EXHIBIT B

Patent Application
Client Docket No. **SUN04-0078-EKL**
(017.0379.US.UTL)

SYSTEM AND METHOD FOR DYNAMIC PRELOADING OF CLASSES THROUGH MEMORY SPACE CLONING OF A MASTER RUNTIME SYSTEM PROCESS

Field of the Invention

The invention relates in general to class preloading and, in particular, to a system and method for dynamic preloading of classes through memory space cloning of a master runtime system process.

Background of the Invention

Recent advances in microprocessor design and component integration have enabled a wide range of devices to offer increasingly complex functionality and "soft" features. Soft features include software applications that enhance and customize the operation of a device. These devices include standard computing devices, such as desktop and laptop computers, portable computing devices, such as personal data assistants, and consumer devices, such as cellular telephones, messaging pagers, gaming consoles, and set top boxes. Most devices now include an operating system to support the soft features and other extensions.

The increased capabilities offered by these software-upgradeable devices have also created certain user expectations. Often, users are not technically savvy and are intolerant of performance compromises occasioned by architectural challenges, such as slow or inconsistent application performance. Similarly, users generally expect to be able to access a host of separate applications, which are implemented at the system level through multitasking. For users, widely available software applications assure a positive experience through consistency and increased exposure across multiple platforms. However, for software developers, engineering software applications for disparate computing platforms

What is claimed is:

1. A system for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:

a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code;

a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process; and

a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process.

2. A system according to Claim 1, further comprising:

a cache checker to determine whether the instantiated class definition is available in a local cache associated with the master runtime system process.

3. A system according to Claim 2, further comprising:

a class locator to locate the source definition if the instantiated class definition is unavailable in the local cache.

4. A system according to Claim 1, further comprising:

a class resolver to resolve the class definition.

5. A system according to Claim 1, further comprising:

at least one of a local and remote file system to maintain the source definition as a class file.

6. A system according to Claim 1, further comprising:

a process cloning mechanism to instantiate the child runtime system process by copying the memory space of the master runtime system process into a separate memory space for the child runtime system process.

7. A system according to Claim 1, further comprising:

a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

8. A system according to Claim 1, wherein the master runtime system process is caused to sleep relative to receiving the process request.

9. A system according to Claim 1, further comprising:

a resource controller to set operating system level resource management parameters on the child runtime system process.

10. A system according to Claim 1, wherein the object-oriented program code is written in the Java programming language.

11. A system according to Claim 10, wherein the master runtime system process and the child runtime system process are Java virtual machines.

12. A method for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:

executing a master runtime system process;

obtaining a representation of at least one class from a source definition provided as object-oriented program code;

interpreting and instantiating the representation as a class definition in a memory space of the master runtime system process; and

cloning the memory space as a child runtime system process responsive to a process request and executing the child runtime system process.

13. A method according to Claim 12, further comprising:

determining whether the instantiated class definition is available in a local cache associated with the master runtime system process.

14. A method according to Claim 13, further comprising:

locating the source definition if the instantiated class definition is unavailable in the local cache.

15. A method according to Claim 12, further comprising:

resolving the class definition.

16. A method according to Claim 12, further comprising:

maintaining the source definition as a class file on at least one of a local and remote file system.

17. A method according to Claim 12, further comprising:

instantiating the child runtime system process by copying the memory space of the master runtime system process into a separate memory space for the child runtime system process.

18. A method according to Claim 12, further comprising:

instantiating the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process; and

deferring copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

19. A method according to Claim 12, further comprising:

causing the master runtime system process to sleep relative to receiving the process request.

20. A method according to Claim 12, further comprising:

setting operating system level resource management parameters on the child runtime system process.

21. A method according to Claim 12, wherein the object-oriented program code is written in the Java programming language.

22. A method according to Claim 21, wherein the master runtime system process and the child runtime system process are Java virtual machines.

23. A computer-readable storage medium holding code for performing the method according to Claim 12.

24. An apparatus for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:

means for executing a master runtime system process;

means for obtaining a representation of at least one class from a source definition provided as object-oriented program code;

means for interpreting and means for instantiating the representation as a class definition in a memory space of the master runtime system process; and

means for cloning the memory space as a child runtime system process responsive to a process request and means for executing the child runtime system process.